Rellenar estos cuadros:

Nombre:	
C.I.:	

TECNÓLOGO EN INFORMÁTICA

ESTRUCTURAS DE DATOS Y ALGORITMOS

Examen

11 de Agosto de 2008

(Son 3 carillas)
Tiempo Total: 2:30 hs.

Ejercicio 1 (30 puntos)

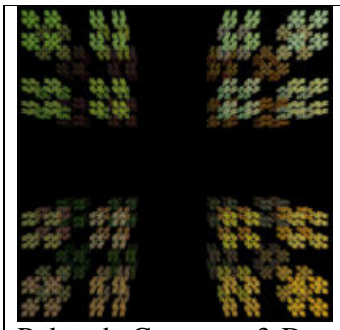
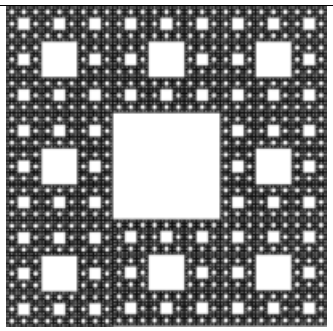
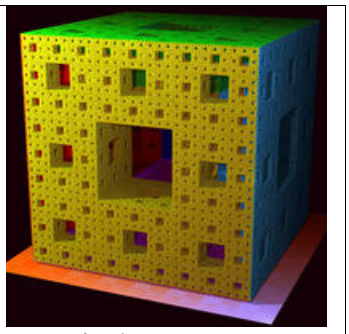
El **Conjunto de Cantor**, llamado así por ser introducido por Georg Cantor en 1883, es un destacado conjunto fractal en el intervalo real [0, 1].

Se construye de modo recursivo dando los siguientes pasos:

- El primer paso es tomar el intervalo [0, 1].
- El segundo paso es quitarle su tercio interior, es decir el intervalo abierto (1/3; 2/3).
- El tercero es quitar a los dos segmentos restantes sus respectivos tercios interiores, es decir los intervalos abiertos (1/9; 2/9) y (7/9; 8/9).
- Así sucesivamente…

La figura muestra las siete primeras etapas:

En dos o más dimensiones se pueden generar figuras como las siguientes:

 Polvo de Cantor en 3-D	 Alfombra de Sierpinski	 Esponja de Menger

Queremos dibujar la figura fractal producida por el conjunto de Cantor y para ello contamos con una función llamada *quitarIntervalo(Figura f, float a, float b)* que remueve el intervalo abierto (a,b) de una Figura f.

Se pide:
Escribir un seudo-código recursivo que dibuje (sobre una figura *f* pasada por parámetro) el conjunto de Cantor para *cant_iter* cantidad de iteraciones (también pasada por parámetro), que utilice la función *quitarIntervalo(Figura f, float a, float b)* para dibujarla.

No se preocupe por pasar parámetros por valor o referencia, suponga que no necesita aclararlo.

Sugerencias:

- Note como en cada intervalo el segmento a quitar está delimitado por:

$$\left(\inf + \left(\frac{1}{3^i}\right), \sup - \left(\frac{1}{3^i}\right)\right)$$

  Donde ***i*** es el paso del algoritmo, ***inf*** se corresponde con el borde inferior del segmento actual, y ***sup*** es el borde superior del mismo segmento.

- Escriba el cabezal de la función recursiva de la siguiente forma:
  conjuntoCantor(Figura f, float max, float min, int paso, int cant_ iter)

Solución

```
void conjuntoCantor(Figura f, float max, float  min, int  paso, int cant_ iter){
   if(paso <= cant_iter)
        float a = min + (1/(3^paso);
        float b = max - (1/3^paso);
        conjuntoCantor(f, max, b, paso + 1, cant_ iter);
        quitarIntervalo(f, a, b);
        conjuntoCantor(f, a, min, paso + 1, cant_ iter);
   }
}
```

Ejercicio 2 (40 puntos)

a) (15 puntos)
Escribir un módulo de definición del TAD árbol binario de cardinales conteniendo el conjunto mínimo de funciones constructoras, de predicado y selectoras.
b) (25 puntos)
Implementar las siguientes operaciones, en forma recursiva, en base al TAD definido en la parte anterior
i. Max: Dado un árbol binario no vacío, retorna el elemento natural más grande
ii. CantHijosIzq: Dado un árbol binario, retorna la cantidad de nodos que tiene hijo izquierdo no vacío.

Ejercicio 3 (30 puntos)

Dadas las siguiente definición de tipo y las definiciones de operaciones funcionales sobre lista de naturales (LNat):

```
typedef unsigned short natural;

LNat Null();
/* Crea la lista vacía */
LNat Cons(natural x, LNat l);
/* Inserta un elemento al principio de la lista */
bool Empty(LNat l);
/* Retorna TRUE si la lista está vacía */
natural Head( LNat l);
/* Retorna, si la lista no es vacía, el primer elemento de la lista */
LNat Tail(LNat l);
/* Retorna, si la lista no es vacía, la lista sin su primer elemento */
```

a)Implementar la siguiente función recursivamente:
BorrarUltimo: Recibe una lista y retorna una lista con el mismo contenido que la lista de entrada pero sin el último elemento. Precondición: la lista parámetro no es vacía.

b) Implementar la siguiente función recursivamente:
ElimPares: Recibe una lista y retorna una lista con el mismo contenido que la lista de entrada pero sin los elementos pares. Precondición: la lista parámetro no es vacía.

- **LNat ElimPares(LNat l);**